Subject: Re: renaming a variable without making a copy
Posted by David Fanning on Tue, 08 Dec 2009 22:39:40 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> In article <MPG.2587fff16b170da398968a@news.giganews.com>,
> David Fanning <news@dfanning.com> wrote:
>
>> newName = Temporary(oldName)
>
> Can anyone explain to me what TEMPORARY actually does? The documentation
> says
>
> The TEMPORARY function returns a temporary copy of a variable, and sets
> the original variable to "undefined".
>
> which makes no sense to me at all. Doesn't making a "temporary copy
> of a variable" occupy memory? Perhaps I am confused by the use of the name
> "TEMPORARY".
>
> My concept of an IDL variable (which could easily be wrong) is: some
> metadata that describes the variable (what you get with the SIZE function)
> and the actual data that comprises the variable. These things could be
> in different places in memory, and the metadata could contain, for example,
> a pointer to the actual data. Most of the time, I don't need to know.
>
> Does TEMPORARY wipe out the old metadata (replacing it with
> "undefined") and create new "unnamed" metadata that points to the data part
> of the destroyed variable?
>
> The example in the Docs is not very revealing.

Here is how I wave my hands around this when explaining it
in an IDL class. Remember, I am speaking metaphorically here.
I have *no* idea what actually happens. ;-)

You are right, a variable in IDL is composed of some metadata,
one part of which is the variable's name, and some machine
memory, where the variable lives. I like to say the variable
is "attached" to the machine memory. When you issue a command
like this:

```
newVar = Temporary(oldVar) + 1
```

You are saying to IDL, "Take that machine memory that is attached
to oldVar and temporarily use it to perform whatever operation
you are doing." Then, when you are finished, make another variable,

newVar, and attach this temporary memory permanently to this variable. In IDL there is a rule that only one variable at a time can be permanently attached to machine memory, so the act of attaching this memory to newVar is to remove it from oldVar. A variable that has no machine memory attached to it is, by definition, an undefined variable.

> Why does
>
> A = TEMPORARY(A) + 1
>
> use less memory than
>
> A = A + 1
>
> I suppose there is a good reason that the latter example "creates a new
> array for the result of the addition, places the sum into the new array,
> assigns it to A, and then frees the old allocation of A", although it
> just seems to me like the interpreter is being obtuse.

I'm sure there is a good reason. And if I think about it long enough, I'm sure it will come to me. Meantime, you may have to take it on faith that IDL just works that way. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")